Tentamen 1td317

Kevin Rasmusson
2022-08-28
0104144092

Generell kommentar
Ber om ursäkt för versaler där det ej skall vara (blir så när man skriver i google docs hehe). Ger väl en bra känsla av "coding-interview" om inte annat. Får även be om ursäkt för blandningen mellan svenska och engelska, i farten (under kodningen) blir kommentarerna på engelska medan de jag lägger till efter har varit både svenska och engelska. Allt gott!

Uppgift 1

Kommentar: En relativt generell uppgit som gick ut på att implementera en klass enligt specifikation. Har namn på engelska eftersom det är så jag föredrar att döpa mina variabler, men behöll iallafall klassnamnet på svenska om ni hade något main program ni tänkte testa med.

```
// hyrbil.h
#ifndef HYRBIL_H
#define HYRBIL_H

class Hyrbil {
private:
   std::string brand;
   std::string model;
   std::string color;
   int year;
   bool rented;
   bool electric; // true if electric, false if other
   int kilometer_counter;
public:
   Hyrbil();
   ~Hyrbil();
   Std::string getBrand();
   Std::string getModel();
   Std::string getColor();
   Int getYear();
   Bool isRented();
   Bool isElectric();
   Int get_kilometer_counter();
   void printInfo(std::ostream &stream);
   int calcDistance(int km);
   void returnCar(int km);
};
```

```
#endif // HYRBIL_H

// hyrbil.cpp

#include <iostream>
#include ”Hyrbil.h”

using namespace std;

Hyrbil::Hyrbil()
: brand(””), model(””), color(””), year(0), rented(false), electric(false), kilometer_counter(0)
{}

Hyrbil::~Hyrbil()
{}

Hyrbil::Hyrbil(string brand, string model, string color, int year, bool rented, bool electric, int
kilometer_counter)
: brand(brand), model(model), color(color), year(year), rented(rented), electric(electric),
kilometer_counter(kilometer_counter)
{}

string Hyrbil::getBrand() {
	return this->brand;
}

string Hyrbil::getModel() {
	return this->model;
}

string Hyrbil::getColor() {
	return this->color;
}

int Hyrbil::getYear() {
	return this->year;
}

bool Hyrbil::isRented() {
	return this->rented;
}

bool Hyrbil::isElectric() {
	return this->electric;
}

int Hyrbil::get_kilometer_count() {
```

```
	return this->kilometer_count;
}

Void Hyrbil::printInfo(ostream &stream) {
	stream << “Brand: “ << this->brand << “\nModel: “ << this->model << “\nColor: “ <<
this->color << “\nYear: “ << this->year << “\nRented: “ << this->rented << “\nElectric: “ <<
this->electric << “\nKilometer Count: “ << this->kilometer_count << endl;
}

/**
*Tolkar det som att km är den nya mätarställningen, och att vi får diff då.
*/
int Hyrbil::calcDistance(int km) {
	return (km - this->kilometer_count);
}

void Hyrbil::returnCar(int km) {
	this->isRented = false;
	this->kilometer_count = this->kilometer_count + km;
}
```

Uppgift 2

```
// main.cpp
#include <iostream>
#include <fstream>

Using namespace std;

int NUMBER_OF_LETTERS = 27; // använder global för att kunna ändra till åäö också

int main() {
	// array for letters
	int doubleCounter[NUMBER_OF_LETTERS] = {0};
	ifstream is(“fil.txt”);
	char prev = ‘’;
	char temp = ‘’;
	while(!is.eof()) {
		is >> temp;
		if (isAlpha(temp)) { // only operate on if alpha
			if ((int) temp > 90) temp = temp - 32; // convert all to capital letters
			int index = (int) temp - 65; // shift index 65, A = 0, B = 1 etc
			if(temp == prev) { // will fix triplets too, two loops, two counts
				if (index < NUMBER_OF_LETTERS && index > 0) {
					doubleCounter[index]++;
				}
```

```
                        }
                }
                prev = temp;
        }

        // print
        for (int i = 0; i < NUMBER_OF_LETTERS; i++) {
                if (doubleCounter[i] != 0) {
                        // (char) (i + 65) ger bokstaven i versal till utskrift (alphanum table)
                        cout << (char) (i + 65) << " : " << doubleCounter[i] << endl;
                }
        }
}
```

Uppgift 3

Kommentar: Valde att undvika array enligt order, se kommentarer för ytterligare implementationsdetaljer. I ord, kollar datumen, jämför som integers, om första filens är mindre cout:as den till output filen, och tvåan går tillbaka till samma avläsning som den var på, medan fin1 avancerar till nästa datum. Fortsätter tills en fil är slut, då gås resten av filen som har något kvar igenom.

```
// main.cpp
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

int main() {
   ifstream fin1, fin2;
   ofstream fout;
   int n1, n2;
   string s1, s2;
   int place1 = 0;
   int place2 = 0;

   fin1.open("temperatur1.dat");
   fin2.open("temperatur2.dat");
   fout.open("temperatur.dat");

   while(fin1 >> n1 && fin2 >> n2) // kollar om det finns något att hämta i vardera av filerna
   {
      if(n1 < n2)
      {
         double temp; // för att hålla temperaturen i en double, gäller samtliga instanser av
denna "kodsnutt"
```

```
            fin1 >> temp;
            fout << n1 << " " << temp << endl;
            fin2.seekg(place2); // flyttar cursor till
        } else {
            double temp;
            fin2 >> temp;
            fout << n2 << " " << temp << endl;
            fin1.seekg(place1);
        }

        place1 = fin1.tellg();
        place2 = fin2.tellg();
    }

    // dessa två kommande while loops är för att “tömma” den filen som fortfarande har data i sig som gjorde att vi hoppade ur den första
    while(fin1 >> n1) {
        double temp;
        fin1 >> temp;
        fout << n1 << " " << temp << endl;
    }

    while(fin2 >> n2) {
        double temp;
        fin2 >> temp;
        fout << n2 << " " << temp << endl;
    }

    // close
    fin1.close();
    fin2.close();
    fout.close();
}
```

Uppgift 4

Kommentar: Här ligger allt i en main.cpp fil istället för uppdelningen jag gjorde i uppgift 1, mestadels för att jag faktiskt ville ta mig tiden att kompilera detta utan att behöva kriga med att använda flera filer. Hoppas det är ok, hade även flyttat om ordningen lite om jag faktiskt hade bestämt mig för att ha allt i en fil, dvs, klassdeklarationer på en plats, implementationen längre ned, och main emellan. Men nu får det ligga som det ligger.

```
#include <iomanip>
#include <iostream>
#include <string>

using namespace std;
```

```
class Person
{
private:
  string namn;
  string nummer;
  string extraheraSiffror(const string &nummer);

public:
  Person();
  Person(string ny_namn, string ny_nummer);
  string getNamn();
  string getNummer();
  void skrivUt(ostream &out);
  bool sok(const string& key);
};

Person::Person() : namn(""), nummer("") {}

Person::Person(string ny_namn, string ny_nummer)
  : namn(ny_namn), nummer(ny_nummer) {}

string Person::getNamn()
{
  return namn;
}

string Person::getNummer()
{
  return nummer;
}

void Person::skrivUt(ostream &out)
{
  out << left << setw(40) << namn << " " << nummer << endl;
}

string Person::extraheraSiffror(const string &nummer)
{
  string bara_siffror;
  for (unsigned int i = 0; i<nummer.size(); ++i)
    if (isdigit(nummer[i]))
      bara_siffror.push_back(nummer[i]);
  return bara_siffror;
}

bool Person::sok(const string& key)
{
```

```
    string extract_key = extraheraSiffror(key);
    string bara_siffror = extraheraSiffror(nummer);
    // find key in part of namn or bara_siffror
    if (namn.find(extract_key) != string::npos || bara_siffror.find(extract_key) != string::npos)
        // kikar om del av antingen namnet eller av telefonnummer
        return true;
    else
        return false;
}

class TelefonLista {
private:
    int personCounter;
    Person *personer;
public:
    TelefonLista();
    ~TelefonLista();
    void laggTill(const Person &person);
    void hittaOchSkrivUt(const string &key);
    TelefonLista &operator=(const TelefonLista &lista);
};

TelefonLista::TelefonLista() : personCounter(0), personer(NULL) {}

TelefonLista::~TelefonLista()
{
    delete[] personer;
}

void TelefonLista::laggTill(const Person &person)
{
    Person *nyPerson = new Person[personCounter + 1];
    for (int i = 0; i < personCounter; ++i)
        nyPerson[i] = personer[i];
    nyPerson[personCounter] = person;
    delete[] personer;
    personer = nyPerson;
    ++personCounter;
}

void TelefonLista::hittaOchSkrivUt(const string &key)
{
    int found = 0; // simpel counter för om vi hittat någon
    for (int i = 0; i < personCounter; ++i)
        if (personer[i].sok(key)) {
            personer[i].skrivUt(cout);
            ++found;
        }
```

```
    if (found == 0) cout << "Kunde inte hitta " << key << " i telefonlistan!" << endl;
}

// for deep copy
TelefonLista & TelefonLista::operator = (const TelefonLista &other) {
    if (this != &other) {
        delete[] personer;
        this->personCounter = other.personCounter;
        for (int i = 0; i < personCounter; i++) {
            this->personer[i] = other.personer[i];
        }
    }
    return *this;
}


int main() {
    TelefonLista lista;
    lista.laggTill(Person("Andersson, Adam", "070 1234567"));
    lista.laggTill(Person("Andersson, Magdalena", "076-234 5678"));
    lista.laggTill(Person("Anka, Kalle", "+46-18-171 932"));
    lista.laggTill(Person("Duck, Donald", "+1-732 1938 3234"));
    lista.laggTill(Person("Johansson, Johan", "+46-72-325 42 42"));
    lista.laggTill(Person("Jonsson, Ibrahim", "+46-72-843 41 52"));
    lista.laggTill(Person("Ibrahimovic, Anders", "+39-471-268 52 35"));
    lista.laggTill(Person("Ibrahimovic, Anna", "0732444985"));
    lista.laggTill(Person("Reinfeldt, Fredrik", "0701-419 273"));
    lista.laggTill(Person("von Anka, Joakim", "+45-533 30 43 940"));

    while(true)
        {
        cout << "Ange person eller nummer som ska hittas (sluta med 0): ";
        string input;
        getline(cin, input);
        if (input == "0")
            break;
        lista.hittaOchSkrivUt(input);
        }

    return 0;
}
```